Modelling the Diffusion of a One Dimensional Rod with Physics Informed Neural Networks

Leah Hansen
Department of Physics, University of Oslo
(Dated: March 1, 2025)

A recent interest in the field of Machine Learning is the use of Physics Informed Neural Networks (PINN). PINNs are commonly used for solving ODEs and PDEs, which can often be used to model real-world systems, [1][2]. In our study, we have made a PINN to solve the one-dimensional diffusion equation with a diffusion coefficient $D = 1$. Our system is a one-dimensional rod of length $L = 1$, where we attempt to model the temperature diffusion during a total time $T = 1$. We have compared the PINN model to the more common method of using Forward Time Centered Space (FTCS) Euler. We tried several activation functions, hidden nodes and layers to find our best PINN model. Our best PINN model produced an RMSE of 2.27×10^{-5} for the whole simulation time. In comparison, the best numerical model using the FTCS method for a Δx=0.01 gave an RMSE of 5.44×10^{-6} for the timepoint 0.13, while for the same Δx and timepoint, our best PINN had an RMSE of 2.00×10^{-5}. For $\Delta x = 0.1$ and t = 0.12 the best PINN model had an RMSE of 8.11×10^{-5} and the FTCS model had an RMSE of 2.84×10^{-3}. This was for a 10×10 grid in the PINN model. In the future, we would like to expand our PINN model for diffusion coefficients other than 1.

I. INTRODUCTION

Physics Informed Neural Networks (PINNs) have in recent years been gaining popularity, particularly for solving nonlinear Partial Differential Equations (PDEs) [3]. PINNs are a type of Neural Network (NN), which define the loss function in a way that upholds the physics of the system when the loss function is minimized. This means that alongside the predictive strengths of NNs we get to ensure that our system remains physically sensible through the boundaries set.

In our study we have attempted to use PINNs to solve the one-dimensional diffusion equation with a diffusion coefficient of 1. We have solved the PDE in a unitless system, where we can imagine the diffusion to represent the diffusion of heat or other things. Our goal is to demonstrate the usefulness of machine learning in solving these types of problems, as the further application of PINNs on PDEs can have many interesting real-life applications. For instance, in nonlinear PDEs and simulations of chaotic systems, such as the Kuramoto–Sivashinsky equation modeled in Wang et al. [2].

However, we have started small, and in our study, we will only focus on the one-dimensional diffusion equation. We implemented the commonly used Forward Time Centered Space Euler scheme for the diffusion equation and compared the best solution of the FTCS scheme and the PINN model.

A short introduction to key concepts and methods are given in Section II, where the procedure of our numerical experiments are given in Section II D. Our results and the respective discussions are presented in Section III. We conclude our discussion in Section IV

[1] https://github.com/leahelha/FYS-STK4155/tree/main/Project2

II. FORMALISM

A. Setup

In this study, we will model the diffusion of a rod of length $L = 1$ by solving the one-dimensional diffusion equation for a total time $T = 1$. This diffusion could represent, among other things, the temperature diffusion of the rod.

The 1D diffusion equation is given by

$$D\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, \tag{1}$$

where $u(x,t)$ can be f.ex the temperature of the rod, x represents the position on the rod and t represents time. We have defined $x \in [0,1]$. D is the diffusion coefficient, which in our setup has been set to $D = 1$.

In our model, we have the following boundary conditions in space

$$u(0,t) = 0, \quad t \geq 0$$
$$u(L,t) = 0, \quad t \geq 0.$$

For time we have the initial condition

$$u(x,0) = \sin(\pi x), \quad 0 \leq x \leq L$$

The analytical solution of the 1D diffusion equation is given by

$$u(x,t) = \sin(\pi x)e^{-\pi^2 t}. \tag{2}$$

The derivation of this solution can be found in the appendix, Section V A.

We will use the analytical solution to evaluate our numerical models of the system.

B. Numerical methods

In our study, we will be solving the 1D diffusion equation for our setup in two ways. One method is implementing the Forward Euler method also called Forward Time Centered Space (FTCS). The other method is implementing a Neural Network (NN) using **Pytorch** [4]. We will compare the two methods.

1. *Forward Time Centered Space Euler*

When modeling the diffusion equation without using machine learning, we implemented the finite difference method called Forward Time Centered Space Euler. This is a commonly used scheme for the one-dimensional diffusion equation and is an explicit scheme in time.

At i steps and j time-steps the time t_j and position x_i is given by

$$t_j = j\Delta t \;\; j \geq 0$$
$$x_i = i\Delta x \;\; 0 \leq i \leq N,$$

where N is the length of x.

The numerical approximation to the derivative of $u(x,t)$ with respect to t becomes

$$u_t \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t) - u(x_i,t_j)}{\Delta t},$$

with an approximation error of $\mathcal{O}(\Delta t)$.

Similarly, the numerical approximation of the spatial double derivative for x_i and t_j is given as

$$u_{xx} \approx \frac{u(x_i+\Delta x_i,t_j) - 2u(x_i,t_j) + u(x_i-\Delta x,t_j)}{\Delta x^2},$$

where the approximation error is of $\mathcal{O}(\Delta x^2)$.

We may simplify the terms to

$$u_t \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t}$$

and

$$u_{xx} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta t}.$$

This allows us to write out discretized 1D diffusion equation as,

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta t}.$$

The solution is only stable if the condition $\alpha \leq \frac{1}{2}$ is met, where $\alpha = \frac{\Delta t}{\Delta x^2}$. Thus, our scheme is

$$u_{i,j+1} = \alpha u_{i-1,j} + (1-2\alpha)u_{i,j} + \alpha u_{i+1,j}. \tag{3}$$

The algorithm we will be using is explained more succinctly in Alg.(1). For further reading on discretization of the 1D diffusion equation and implementing FTCS see Chapter 10 in FYS4150 lectures by Morten Hjort-Jensen[5].

Algorithm 1 FTCS Euler for 1D Diffusion Equation with D=1

Given: PDE $\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}$
Boundary conditions: $u(0,t) = 0$, $u(L,t) = 0$
Initial condition: $u(x,0) = \sin(\pi x)$
Discretize: Spatial domain $0 \leq x \leq L$ with N spatial grid points
Discretize: Time domain $t_n = n\Delta t$ with M time steps, total time T
Set: Spatial and temporal step sizes: $\Delta x = \frac{L}{N}$, $\Delta t = \frac{T}{M}$
Initialize: Solution matrix $u_{i,0} = \sin(\pi i \Delta x)$ for $i = 1, 2, \ldots, N-1$
for $n = 1$ to M **do**
  for $i = 1$ to $N-1$ **do**
    $u_{i,j+1} = \frac{\Delta t}{(\Delta x)^2} u_{i-1,j} + (1 - 2\frac{\Delta t}{(\Delta x)^2}) u_{i,j} + \frac{\Delta t}{(\Delta x)^2} u_{i+1,j}$
  Apply boundary conditions: $u_{0,n} = u_{N,n} = 0$
Output: Final solution matrix u

2. *Physics Informed Neural Network*

Physics Informed Neural Networks, popularized by M. Raissi et al. [1], are often used for solving Ordinary Differential Equations (ODEs) and PDEs, [2]. A PINN is a type of NN where the loss function is defined so that the physics of the system is upheld when the loss function is minimized.

The goal of this study is to illustrate the usefulness of machine learning in solving problems like the 1D diffusion equation. A detailed explanation of Neural Networks (NN) and their function is given in Monsen et al. [6]. We will not be going into detail on NNs in this report.

The Universal approximation theorem states that a feedforward NN can approximate any function with one hidden layer at any precision given enough hidden nodes, Cybenko [7] and Hornik [8]. This theorem is the basis of why we can use NNs to solve the diffusion equation.

First, we must define a trial function, $g_t(x)$.

We define

$$g_t(x) = h_1(x) + h_2(x, N(x,P)),$$

where $h_1(x)$ and $h_2(x,N(x,P))$ are functions that ensure that the trial function satisfies the initial conditions (ICs) and boundary conditions (BCs), and (x,P) is the NN with weights and biases P.

The trial function is defined so that we maintain hard constraints on the initial condition and boundary conditions of our given system.

In our study, we have defined the following trial function,

$$g_t(x,t) = (1-t)\sin(\pi x) + x(1-x)tN(x,t,P), \quad (4)$$

where $g_t(x,t)$ and $N(x,t,P)$ are functions of space x and time t.

The loss function serves the purpose of upholding the physics of the PINN. In our case, the loss function is defined as the mean square residual, given by

$$C(x,t,P) = \frac{1}{N_P}\sum\left(\frac{\partial^2 g_t(x,t)}{\partial x^2} - \frac{\partial g_t(x,t)}{\partial t}\right)^2, \quad (5)$$

where N_P is the number of data points defined as $N \times M$ where N is the number of spatial points and M is the number of temporal points. The PINN will attempt to minimize the loss function so that it fulfills the 1D diffusion equation, Eq.(1).

C. Evaluation metrics

We determined the validity of our results using the Mean Squared Error given by

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2, \quad (6)$$

where n is the number of observations, y_i is the true value of the ith observation and $\tilde{y}_i$ is our predicted value for the ith observation.

We will also frequently look at the root of the MSE, the RMSE,

$$RMSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2}. \quad (7)$$

D. Procedure

After finding the analytical solution to our diffusion equation, Eq.(1), we ran the FTCS scheme described in Alg.(1). We found the numerical solution for an early time-point t_1 and a late time-point t_2 for $\Delta x = \frac{1}{10}$ and $\Delta x = \frac{1}{100}$, and compared the numerical and analytical solutions to the specifications.

We initialized our PINN specifying a certain number of nodes and hidden layers, for different activation functions. The setup to our problem is predefined, and we implemented our trial- and loss function as explained in Eq.(4) and Eq.(5) respectively.

In our study, we have decided to test out three different activation functions, **Tanh**, **SiLU**, and **Sigmoid**.

The main purpose of the activation function is to introduce non-linearity between the layers in the NN.

The **Tanh** activation function is given by

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad (8)$$

where z is the output of a layer.

The **Sigmoid** function is given by

$$f(z) = \frac{1}{1+e^{-z}}. \quad (9)$$

The **SiLU** activation function is given by

$$f(z) = \frac{z}{1+e^{-z}}. \quad (10)$$

For our PINN we have used the optimization method ADAM, which is given a general introduction in Monsen et al. [6]. However, in this study, we have implemented a built-in optimizer from the **PyTorch** framework.

In order to find the ideal hyper-parameters we performed a grid search for various nodes, layers, and activation functions. This is done for 5000 epochs with a constant learning rate. When we have found our optimal model we implement an exponential learning rate decay scheduler as suggested by Wang et al. [2], we plot the loss over epochs and study the convergence of the model. The scheduler is a built-in function of **PyTorch**.

Once we have found an optimal PINN model, we compare our results with that of the analytical solution as well as the numerical solution of the FTCS Euler method for the same time points and the same Δx.

III. RESULTS AND DISCUSSION

A. FTCS

In Fig.(1) and Fig.(1) we have plotted the result of the explicit FTCS scheme for a $\Delta x = 0.1$ and $\Delta x = 0.01$ respectively. In each figure, the line in blue indicates the numerical solution for an early time, $t = 0.01$s, and the green solid line indicates the same solution for a later time. In Tab.(II) we see the RMSE values calculated for each solution in each time point.

As expected, the numerical FTCS solution with a $\Delta x = 0.01$ has a much smaller RMSE on average than the FTCS solution with $\Delta x = 0.1$. It's trivial that more spatial points will increase the model's spatial resolution, as is what happens when we decrease Δx.

From Fig.(1) and Fig.(2) we see that $u(x,t)$ in the rod starts relatively high, with a majority of the intensity concentrated in the center of the rod, as time passes we see the intensity even out and spread to the rest of the rod.

TABLE I: RMSE for the FTCS Euler solutions to the 1D diffusion equation, presented in Fig.(1) and Fig.(2). Δt was defined to meet the stability criteria.

Δx	Time point t	$RMSE$
0.1	0.01	3.76×10^{-4}
0.1	0.13	2.84×10^{-3}
0.01	0.01	5.44×10^{-6}
0.01	0.12	4.16×10^{-5}

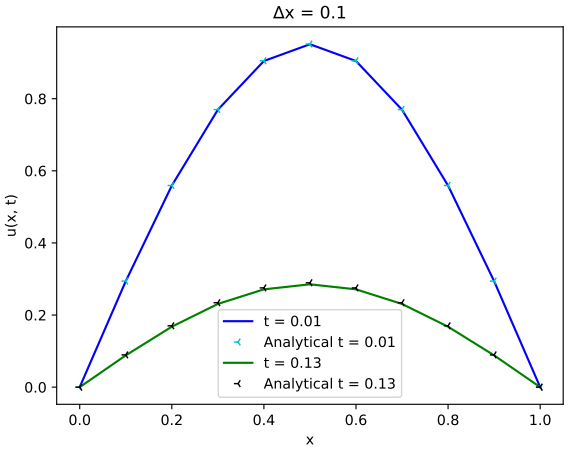

FIG. 1: Plot of the numerical and analytical solution to the diffusion equation for a time $t = 0.01$ and a $t = 0.13$, with $\Delta x = 0.1$. Δt was defined to meet the stability criteria. The method used was the FTCS Euler method. For $t = 0.01$ $RMSE = 3.76 \times 10^{-4}$ and for $t = 0.13$ $RMSE = 2.84 \times 10^{-3}$.

B. PINN

1. *Grid searching*

During our grid search, we found the best PINN model to be the one with the **SiLU** activation function running with 40 hidden nodes and 3 hidden layers. This was performed for 5000 epochs. In figures (3),(4) and (5) we see the result of the grid searches for each activation function's best learning rate.

We note that all the activation functions reach an RMSE at the same order of 10^{-4} with their optimal combinations of the hyper-parameters.

2. *Convergence of best model*

Fig.(6) is the result of plotting the best PINN model with a scheduled exponentially decaying learning rate. In the figure, we see that the model converges at around 24000 epochs. The RMSE for the whole simulation using the PINN model was found to be 2.27×10^{-5} after convergence. Which is a lot higher than it was during our testing phase at 5000 epochs.

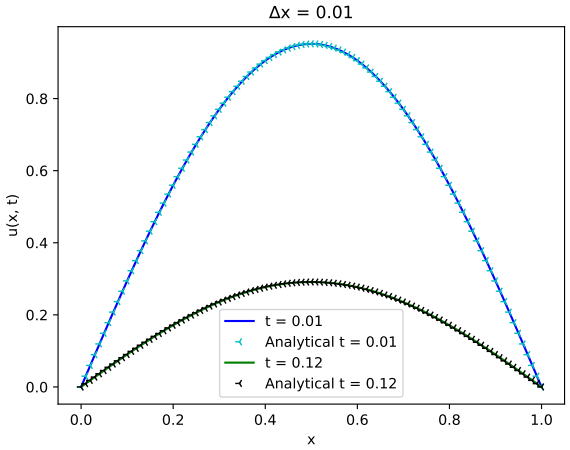

FIG. 2: Plot of the numerical and analytical solution to the diffusion equation for a time $t = 0.01$ and a $t = 0.13$, with $\Delta x = 0.01$. Δt was defined to meet the stability criteria. The method used was the FTCS Euler method. For $t = 0.01$ $RMSE = 5.44 \times 10^{-6}$ and for $t = 0.13$ $RMSE = 4.16 \times 10^{-5}$

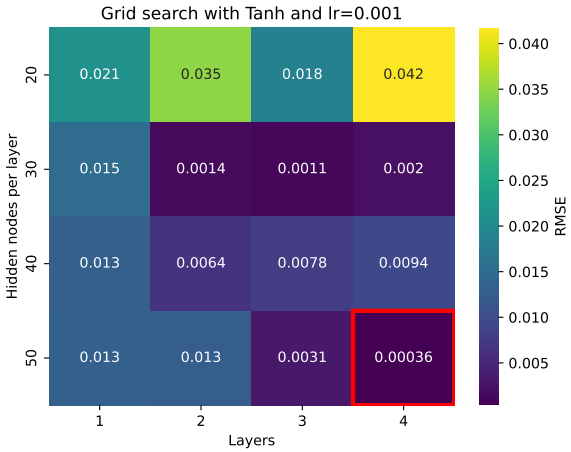

FIG. 3: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is **Tanh**. With this activation function, the learning rate that gave the lowest RMSE was 0.001. See the appendix for the other learning rate results.

C. Comparision

In Fig.(7) and Fig.(8) we see a plot of the best model we made using PINNs, alongside the analytical solution.

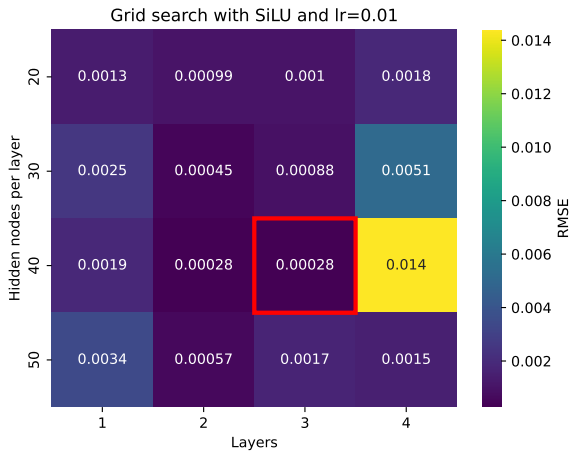

FIG. 4: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is **SiLU**. With this activation function, the learning rate that gave the lowest RMSE was 0.01. See the appendix for the other learning rate results.

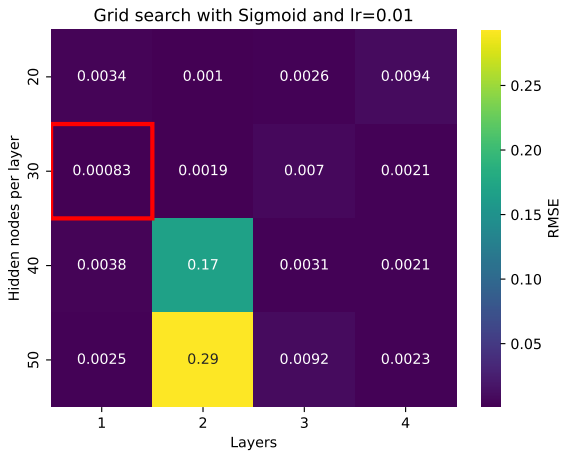

FIG. 5: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is **Sigmoid**. With this activation function, the learning rate that gave the lowest RMSE was 0.01. See the appendix for the other learning rate results.

When comparing Fig.(7) and Fig.(8) with the FTCS solutions in Fig.(1) and Fig.(2), we see that both the PINN model and the FTCS solution do a good job of approximating the analytical solution.

In Tab.(I) we have calculated the RMSE for the two numerical models. Here we are able to actually observe the strengths of the PINN model. In the table, we see that the PINN model consistently has an RMS of an order of 10^{-5} for both Δx. This we can read, means that it outperforms FTCS at a $\Delta x = 0.1$. At $\Delta x = 0.01$ FTCS slightly outperforms the PINN model for a time point $t = 0.01$. However, when we consider that the grid size of the PINN model is only 10×10. At such a small grid, the performance is already really good. It is reasonable to assume that a slightly larger grid would mean that the PINN model could outperform the FTCS solution completely at $\Delta x = 0.01$ as well. Additionally, it is worth noting that the PINN model did not train on upholding the stability criterion. Despite this, it performed really well.

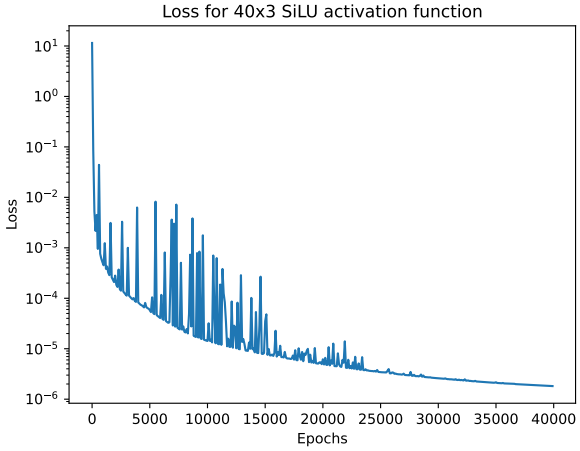

FIG. 6: Plot of loss over epochs for the PINN model with 40 hidden nodes, 3 layers and the activation function **SiLU**. We arrive at a convergence at around 2400 epochs.

TABLE II: RMSE for the FTCS Euler solutions to the 1D diffusion equation, presented in Fig.(1) and Fig.(2). Δt was chosen to meet the stability criteria.

Method for $\Delta x = 0.1$	Time point t	$RMSE$
PINN	0.01	3.48×10^{-5}
PINN	0.12	8.11×10^{-5}
FTCS	0.01	3.76×10^{-4}
FTCS	0.12	2.84×10^{-3}
Method for $\Delta x = 0.01$	Time point t	$RMSE$
PINN	0.01	1.83×10^{-5}
PINN	0.13	2.00×10^{-5}
FTCS	0.01	5.44×10^{-6}
FTCS	0.13	4.16×10^{-5}

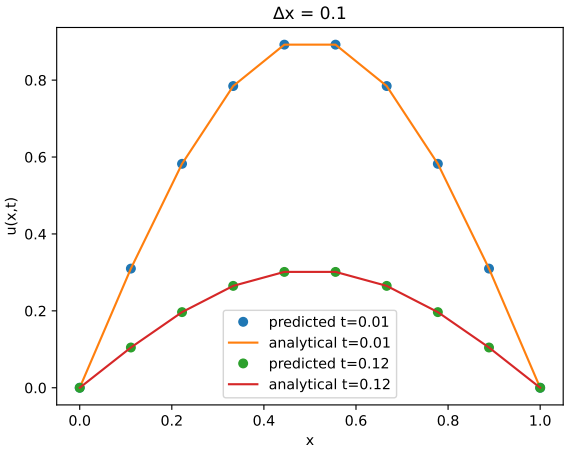

FIG. 7: Plot of the PINN solution of the 1D diffusion equation alongside the analytical solution, for a time $t = X$ and a $t = X$, with $\Delta x = 0.1$.

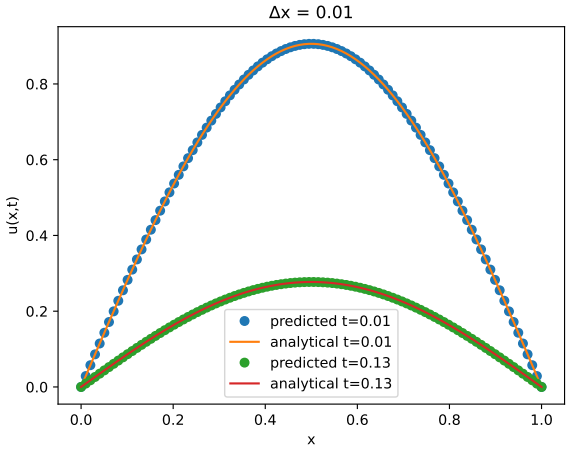

FIG. 8: Plot of the PINN solution of the 1D diffusion equation alongside the analytical solution, for a time $t = X$ and a $t = X$, with $\Delta x = 0.01$.

IV. CONCLUSION

The explicit FTCS scheme was used to solve the 1D diffusion equation, and the results of this are in Fig.(1) and Fig.(2). The simulations were conducted for different spatial resolutions Δx and time points, with corresponding RMSE values summarized in Tab.(II). As anticipated, decreasing Δx improved spatial resolution, resulting in lower RMSE values. We performed a grid search, finding the optimal hyperparameters for the PINN model, including different activation functions, hidden nodes, and layers. From our search we found the optimal PINN model to be the model with 40 hidden nodes, 3 layers, and the activation function **SiLU**. We arrived at a convergence at around 24000 epochs. After convergence, the simulation had an RMSE of 2.27×10^{-5}.

In the end we compared our best PINN model to the model created by the FTCS scheme. The PINN model consistently exhibited excellent accuracy, with RMSE values of the order of 10^{-5}. While FTCS slightly outperformed PINN at $\Delta x = 0.01$ for a specific time point of $t = 0.01$, however, the PINN model is still impressive considering the small grid size used and that the PINN model was not trained on the stability criterion.

In conclusion, the PINN model yielded remarkable results. To further enhance its accuracy and practicality, we could increase the grid size and train it with a diffusion coefficient D other than 1. To reach the most optimal model, we should have conducted a grid search for the PINN after attaining a convergence at 24000 epochs.

[1] M. Raissi, P. Perdikaris, and G. Karniadakis, Journal of Computational Physics **378**, 686 (2019), last accessed: Dec. 18th 2023.

[2] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An expert's guide to training physics-informed neural networks," (2023), last accessed: Dec. 18th 2023, arXiv:2308.08468 [cs.LG].

[3] D. A. Simone Monaco, "Training physics-informed neural networks: One learning to rule them all?" (2023), last accessed: Dec. 18th 2023.
[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019) pp. 8024–8035, last accessed: Dec. 18th 2023.
[5] M. Hjorth-Jensen, *Computational Physics* (University of Oslo, 2015).
[6] I. Monsen, V. H. Hvoslef, and L. Hansen, "Using a feedforward neural network to perform linear and logistic regression,".
[7] G. Cybenko, Mathematics of Control, Signals, and Systems **2**, 303 (1989), last accessed: Dec. 18th 2023.
[8] K. Hornik, Neural Networks **4**, 251 (1991), last accessed: Dec. 18th 2023.

V. APPENDIX

A. Derivation of analytical solution to the 1D diffusion equation

Solving the one-dimensional heat equation

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}$$

Using separation of variables
Assuming that u can be written as a function of x and t separately

$$u(x,t) = x(x)T(t)$$

Substituting this in the PDE

$$\frac{\partial^2}{\partial x^2} X(x)T(t) = \frac{\partial}{\partial t} X(x)T(t)$$

Rearranging, separating the functions according to their variables.

$$\frac{\partial^2 X(x)}{\partial x^2}\frac{1}{X(x)} = \frac{\partial}{\partial t}T(t)\frac{1}{T(t)} = C, \quad C \in \mathbb{R}$$

Since the two sides are equal, we know they must be a constant. Thus,

$$\begin{aligned} & \frac{\partial^2 X(x)}{\partial x^2}\frac{1}{X(x)} = C \\ \rightarrow \quad & \frac{\partial^2 X(x)}{\partial x^2} = CX(x) \\ \rightarrow \quad & \frac{\partial^2 X(x)}{\partial x^2} - CX(x) = 0 \end{aligned}$$

Similarly for $T(t)$ we get

$$\frac{\partial T(t)}{\partial t} - CT(t) = 0$$

Thus we have that

$$\begin{aligned} \frac{\partial^2 X(x)}{\partial x^2} - CX(x) = 0 \\ \frac{\partial T(t)}{\partial t} - CT(t) = 0 \end{aligned}$$

We try setting $C = -p^2$, where p is a new constant, $p \in \mathbb{R}$.
Inserting boundary conditions

$$IC: \quad u(x,0) = \sin(\pi x)$$

$$BC: \quad u(0,t) = 0, \quad u(L,t) = 0$$

This gives:

$$\begin{aligned} X(0)T(t) = 0 \quad \forall t \\ X(L)T(t) = 0 \quad \forall t \end{aligned}$$

This must mean that

$$\begin{aligned} X(0) = 0 \\ X(L) = 0. \end{aligned}$$

We try to find $X(x)$ using that the solution of an ODE of the form

$$X''(x) + p^2 X(x) = 0$$

is given by

$$X(x) = A\cos(px) + B\sin(px)$$

For $x = 0$

$$\begin{aligned} X(0) &= A \cdot 1 + B \cdot 0 = 0 \\ &\Rightarrow A = 0 \end{aligned}$$

For $x = L$

$$\begin{aligned} x(L) &= B\underbrace{\sin(pL)}_{0} = 0 \\ &\Rightarrow \quad pL = n\pi \quad , n \in \mathbb{Z} \\ p &= \frac{n\pi}{L} \end{aligned}$$

We get that

$$\Rightarrow \quad X(x) = B\sin\left(\frac{n\pi}{L}x\right)$$

Now finding $T(t)$.
We write

$$\frac{\partial T(t)}{\partial t} = \dot{T}(t)$$

$$\begin{aligned} \dot{T}(t) + p^2 T(t) &= 0 \\ \dot{T}(t) &= -p^2 T(t) \end{aligned}$$

Since $p = \frac{n\pi}{L}$, we get that

$$\dot{T}(t) + \left(\frac{n\pi}{L}\right)^2 T(t) = 0$$

From here we see that $T(t)$ likely has the solution

$$T(t) = C\exp\left(-p^2 t\right)$$

$$\begin{aligned} u(x,t) &= X(x)T(t) \\ &= B\sin\left(\frac{n}{L}\pi x\right) C\exp\left(-p^2 t\right) \end{aligned}$$

If we rename all constant B, where $B = B + C$, we get

$$u(x,t) = B\sin\left(\frac{n}{2}\pi x\right)\exp\left(-p^2 t\right)$$

We may now attempt to find B, using that the initial condition is $u(x,0) = \sin(\pi x)$.

$$u(x,0) = B\sin\left(\frac{n}{L}\pi x\right)\exp\left(-\left(\frac{n\pi}{L}\right)^2 \cdot 0\right) = \sin(\pi x)$$

For the IC to be satisfied we need

$$B = 1 \text{ and } n = L$$

Thus the solution of $u(x,t)$ is:

$$u(x,t) = \sin(\pi x)\exp\left(-\pi^2 t\right)$$

B. The rest of the grid search results

FIG. 9: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is tanh.

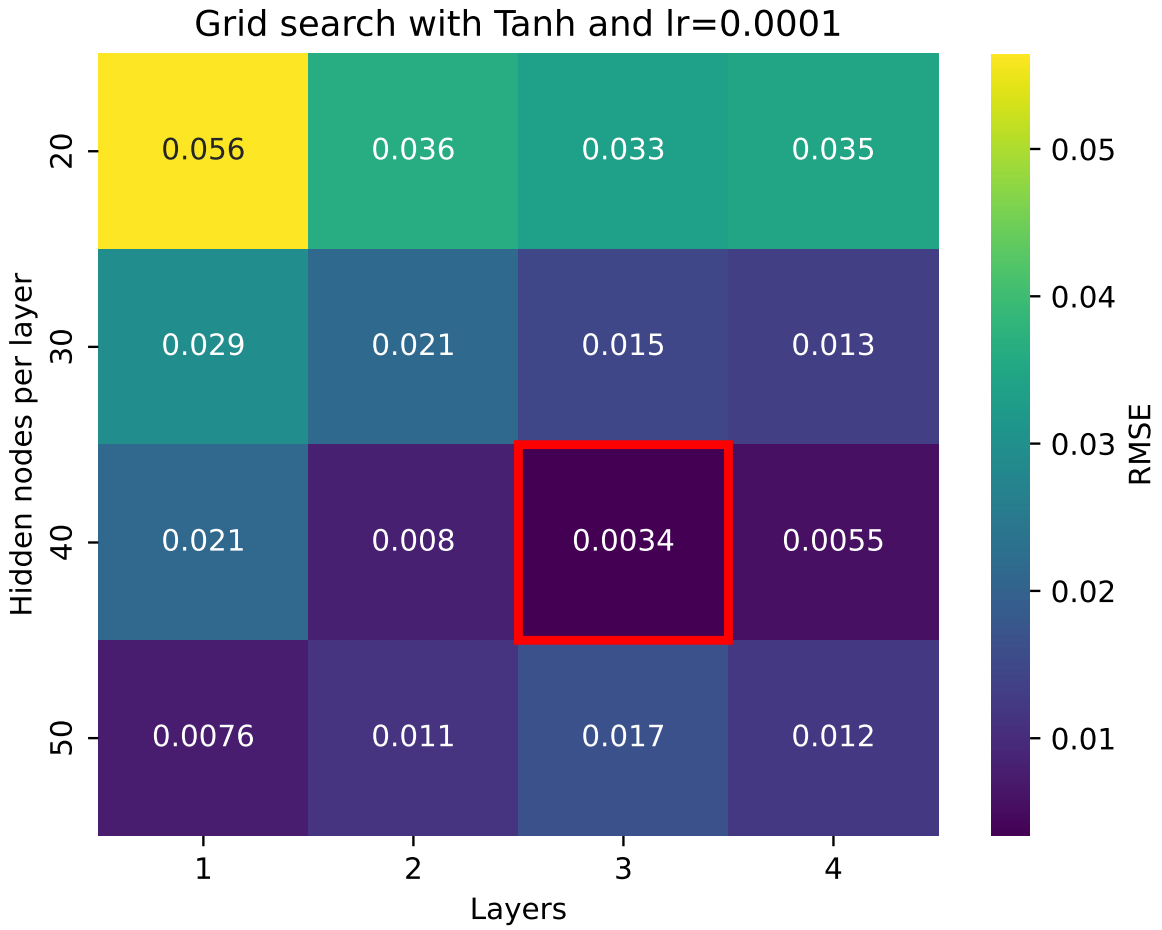

FIG. 10: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is Tanh.

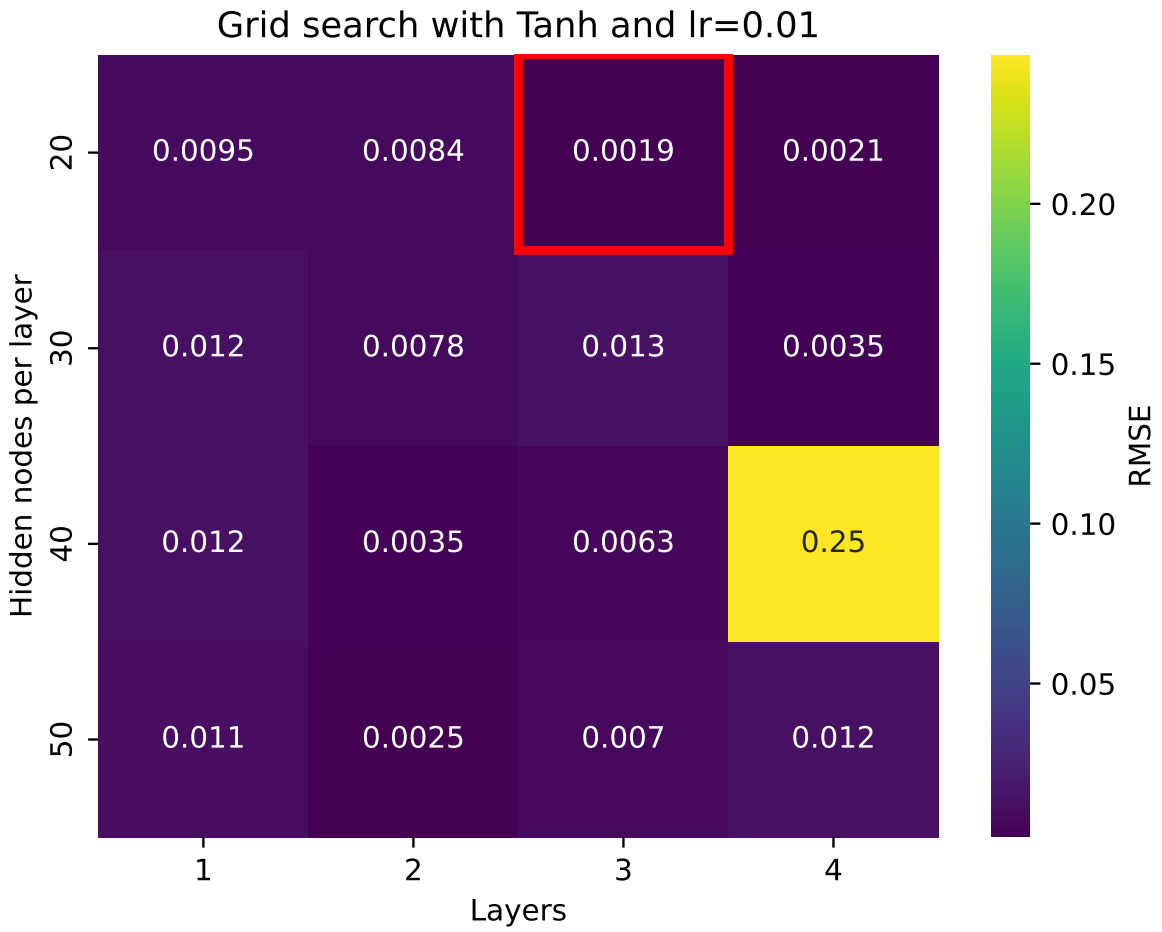

FIG. 11: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is Tanh.

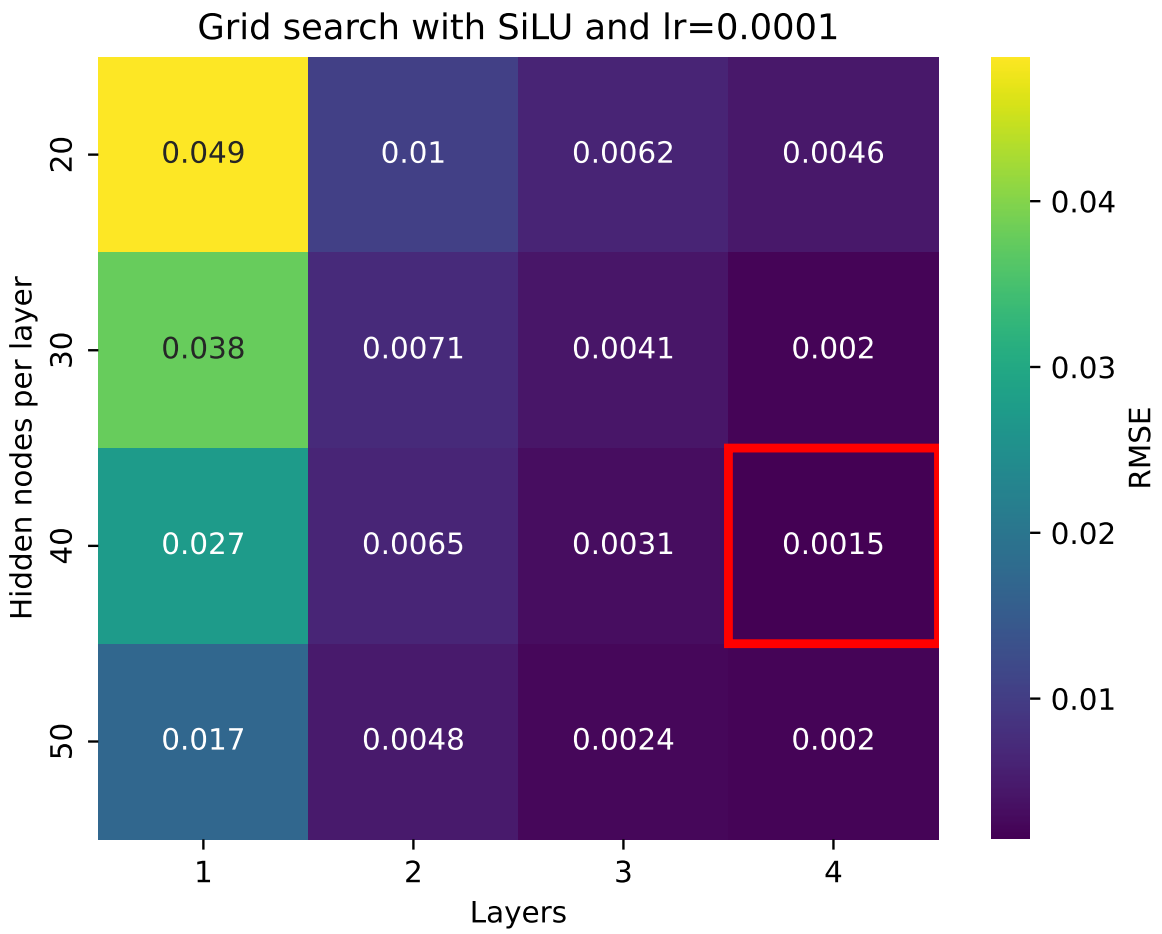

FIG. 12: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is SiLU.

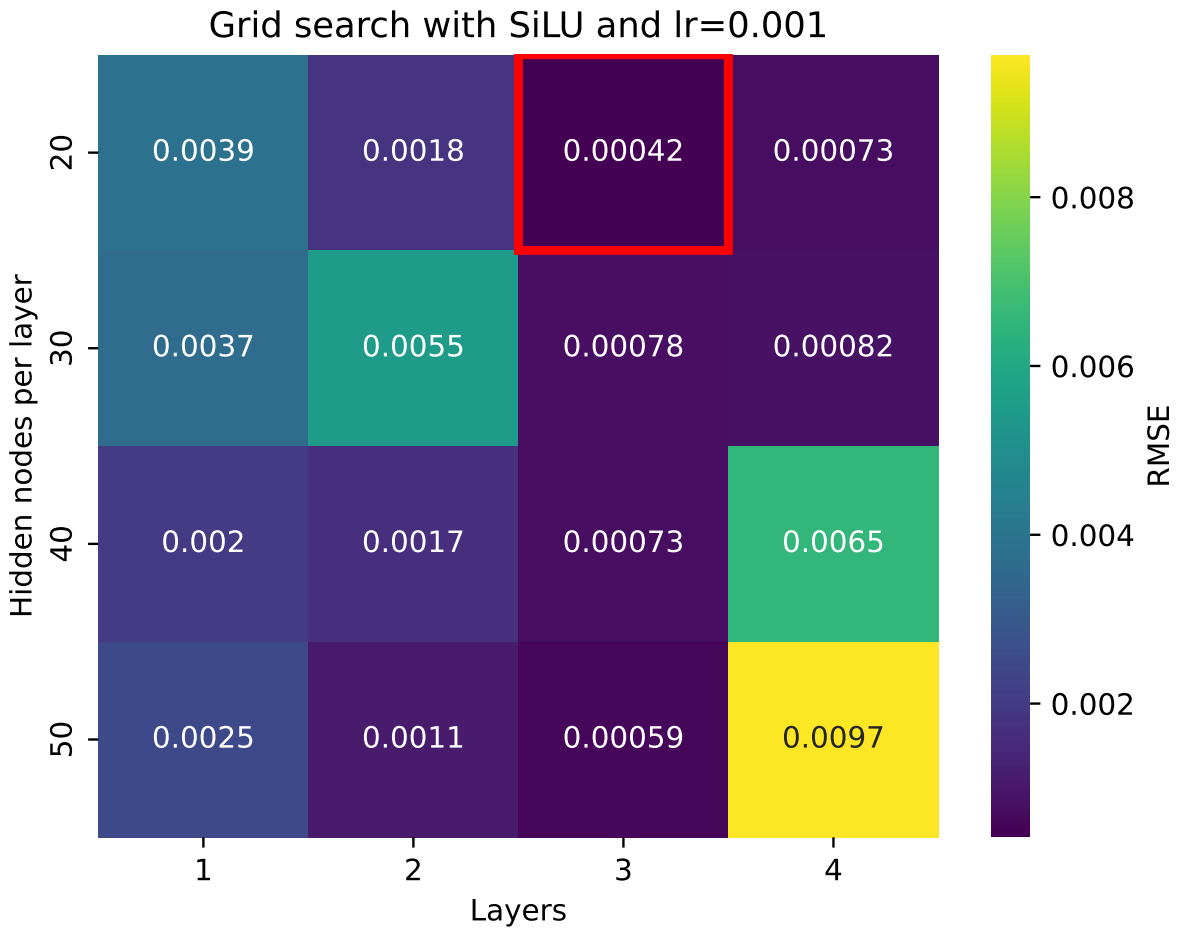

FIG. 13: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is SiLU.

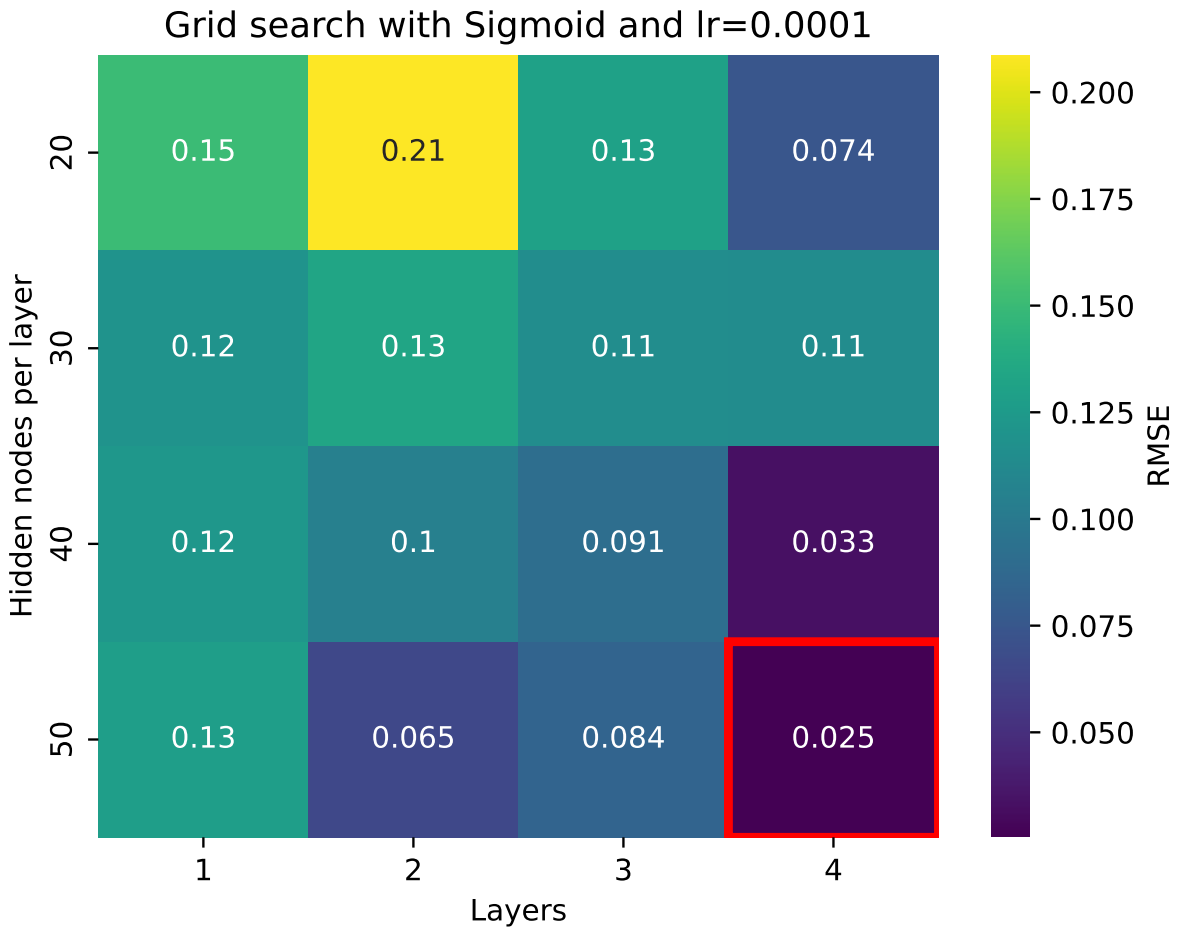

FIG. 14: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is Sigmoid.

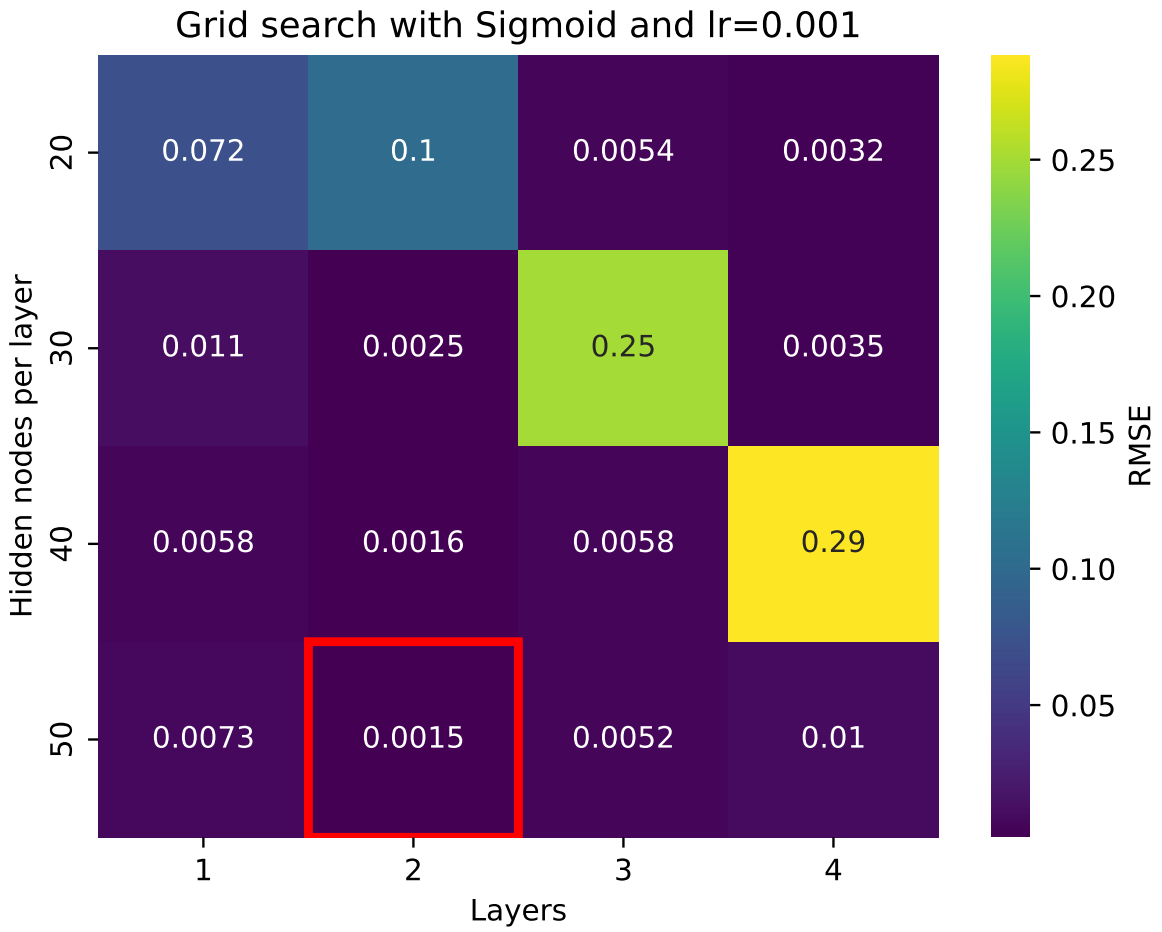

FIG. 15: Grid search for the PINN with hidden nodes in the range [20, 30, 40, 50] and hidden layers, here labeled "layers", in the range [1, 2, 3, 4]. Spatial points $N = 10$ and temporal points $M = 10$, simulation ran for 5000 epochs. The activation function is Sigmoid.